Body Pose Predictions in Triadic Social Interactions

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Rishab Girdhar

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Ju Sun

May, 2021

Acknowledgements

I would like to thank my advisor, Professor Ju Sun for his valuable guidance and support throughout this project. I would also like to thank my committee members Professor Catherine Qi Zhao and Professor Changhyun Choi. A special mention to my colleague and friend Prashanth Kurella without whom this project would not have been possible.

Dedication

To my parents, my brother, my friends especially Suraj, Harshit, Anushree and the Seignious family for helping me survive graduate school and Minneapolis.

Abstract

Human beings are social animals in that they need to socialize with each other to build companionship and thrive alongside other humans. One of the primary characteristics of social interactions is the signals used by people to communicate their thoughts effectively. These include gesturing with their hands, moving around etc.. AI agents or algorithms interacting with humans which we refer to as Social artificial intelligence must learn to interpret and predict these signals in order to use them to interact with other humans successfully. Data-driven approaches have helped make remarkable strides in many artificial intelligence tasks and could similarly help machines learn the body gestures of interacting individuals. We define a framework for predicting these gestures in a triadic social interactions scenario where the humans play a game of haggling and two sellers try to sell their products to a buyer.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

We know that while conversing with people we have a tendency of gesturing with our hands and moving around to convey or emphasize our point. These body gestures are understood by almost everyone and no one is taught about them. People from all cultures and backgrounds gesture because they are elemental for communication. Humans just learn to understand them. Various studies indicate that there is a tight correlation between speech and body gestures and they help to highlight the context of the communication [7]. Speech is a powerful tool but gestures also have their benefits. Our hands help us in talking, thinking and sometimes can divulge information which would be difficult to dissipate through verbal communication especially visuo-spatial information.

Since gestures play such an important role in social interactions, any artificial intelligence(AI) agent aiming to interact genuinely with humans must learn to understand or generate such social signals. Such machines dubbed Social AI in [8] need to have the ability to process and produce the range of social signals observed in human interactions to cooperate with people. One way to build such machines would be to hard code the rules of social behaviors in them. But that is a difficult task to do since non-verbal communication isn't as rigorously researched upon unlike its linguistic counterpart. Such rules are difficult to establish as they come naturally to humans. But as we have seen with the recent advances in computer science, data-driven approaches have proven to be successful in many areas including the great achievements in Natural Language Processing. We have seen the recent GPT-3 [9] model endow machines with ability to

communicate using language. These approaches have not explicitly made use of language structure or grammar rules. This leads us to believe that data driven approaches could help machines in learning non-verbal signals as well.

Researchers have been working hard to develop datasets for various purposes. However, building a data-driven nonverbal communication model is extremely difficult because of the rarity of the data. For language based learning models, there are tons of sources to be found online containing audio and text data in abundance and words can easily be broken down into discrete symbols which can be combined to record the expression of the vocal signals. To "record" non-verbal signals we would need to capture the positions and orientations of people, their body gestures, gaze and expressions and some other characteristics if deemed useful through research. Also, recording these signals for one person would give no idea of the context or the meaning behind the gesture. These semantics cannot be just extracted from millions of videos available online which capture social interactions because they are not available from the raw pixels. Therefore, we need a controlled environment with the correct setup to capture this information. But that also leads to a problem that humans lose their authenticity in gesturing in front of the camera as they may get conscious of the camera. Telling people to perform a gesture while talking would simply make them do actions which would not have the essence of the gesture. Hence we need to tell the people in front of the camera to do such a task which requires gesturing while talking while withholding information of the real nature of the experiment to maintain the authenticity of the social signals.

In their Panoptic Studio [8] Joo et. al used a haggling game(which is discussed in detail in further chapters) to capture the semantic information mentioned above. They captured videos in which three volunteers are engaged in conversation and express their thoughts through words and social signals. The dataset was released online [1] which contains body position, pose, speech data and facial expressions of volunteers along with a baseline model. This information is discussed in details in the further chapters. We use this dataset and attempt to create models which learn to generate social signals of a person based on the social signals of the other two people in the scenario and beat the baseline performance. The further chapters have been divided as follows:

- Chapter 2 covers the related work done in this field. It includes research done to

capture, learn and evaluate models trained to produce human gestures.

- Chapter 3 details the problem statement and the methodologies used.
- In Chapter 4 dataset is discussed in detail and what information from the data was utilized.
- Chapter 5 describes the experiments we conducted and how they were evaluated.
- Chapter 6 presents a final discussion and future work in this field.

Chapter 2

Related Work

2.1 Social Artifical Intelligence

The problem behind Social AI is to create machines capable of interacting with humans realistically using social gestures. This has been presented as "social signal prediction" in the paper by Joo et al [8] as a way to model the dynamics of social signals among interacting individuals in a data-driven manner. They define a scenario which has multiple people interacting in a way which doesn't compromise the integrity of social signals and design a problem statement of predicting these signals. The videos they recorded in their studio with multiple cameras give them the ability to triangulate and provide the 3D locations of keypoints(body joints such as elbows, hands, knees etc..) of human skeletons. They also provide data indicating which of the three interacting individuals is talking for each frame and features which can be used to model the facial expressions of the people. The authors released the dataset along with challenges to predict speaking status, social formation and body gestures of the individuals and provide baseline approaches to these challenges. In our experiments we primarily tackled the problem of predicting body gestures and at the end also tried to find out the speaking status and the potential impact it could have on prediction of body gestures.

In Evonne Ng et al's [10] work on predicting body hand gestures based on the 3D motions of the arm of the speaker. They prove the hypothesis that 3D hand gestures can be synthesized from the motion of the body.

Yoo et al [11] take the interactions with another person whose body can be seen

in the egocentric videos and predict the camera wearer's body in 3D. The key here is that the people visible in the videos are interacting with the person wearing the camera and their social signals are in response to the actions of the first person subject or vice versa.

2.2 Character Movements and Motion Generation

There have been a lot of approaches focusing on predicting or forecasting motion of humans. Recently most approaches have been data-driven and use deep learning to predict 3D pose from input. A lot of motion capture data [Human 3.6M [12], Mocap [13]] is available nowadays thanks to intense research being conducted on this topic worldwide.

Holden el al [14] try to learn the parameters which control the motion of the person. These parameters are represented in a latent dimension as sparse components and combined with different contextual networks to produce realistic motion over the input terrain. They train a convolutional autoencoder to produce the motion representation in the hidden dimension and stack a feedforward network which converts these hidden dimension to high level control parameters. These parameters represent the motion trajectory and movement of hands and legs over the terrain. By making changes in the hidden units they can modify the resulting motion to satisfy any constraints specified.

Recently there have been some approaches making the use of Variational Autoencoders (VAEs) to sample different variations in motion given an input environment. Yan et al [5] model the human motion as a series of motion modes. They refer to a short sequence of human motion over time and state as motion modes and that the trajectory and movement of people over a long time can be thought of as many motion modes sequenced together with transitions in between them. The reason they use VAEs instead of autoencoders is because autoencoders are unable to produce any variations in the output. Their model is able to generate multiple different but viable motion sequences for the same input because of VAEs.

Another approach we see making use of motion VAEs is by Ling et al [6]. They generate human motion by training the autoregressive conditional VAEs also referred to as motion VAEs. We also make use of them and describe the model in detail in further

chapters. By using the latent variables from the autoencoder, they can define the action space for movement and using algorithms for planning or controlling characters on top of them they can generate the motion required. In their paper, they make use of deep reinforcement learning to learn controllers to get movements directed towards a goal.

2.3 Evaluation of Social AI models

With the advent of models working on Social AI, there needs to be certain metrics which could evaluate such models. In the baseline model, the authors used a mean squared error (MSE) to evaluate the output, which, even according to the authors, does not capture the correctness of the produced output as there can be a lot of variation in output and it could still be correct. As there are not defined rules of social interaction, acceptable outputs can be diverse. So other metrics have come up which can compare the essence of the generated results to the ground truth.

2.3.1 Normalized Power Spectrum Score (NPSS)

It was introduced in a paper by Gopalakrishnan et al [15]. This metric was backed by a user study and it tries to capture the change in euler angles in the frequency domain during a movement. The idea behind this metric is that if a person is performing a certain action, we would see a particular distribution in frequency domain and depending on the periodicity and speed of the actions we would see phase shifts in the distribution. The distribution would be more uniform if the motion is non-periodic. This distribution is called the power spectrum and it is obtained using a Fourier transform. The similarity score between two such distributions is referred to as power spectrum score.

2.3.2 Frechet distance

In the work of Yoon et al [16] we see the authors generate gestures of the upper body by providing the context in the form of text, audio and speaker identity. By incorporating multiple modes of context and adversarial training, they are able to generate gestures with varying styles based on speaker identity. They also introduced a new metric for evaluating the generated gestures. They encode the sequence of gestures into a latent dimension and produce a distribution over these dimensions. This distribution is then

compared to the ground truth one using frechet distance. We will use the metric in our evaluation as which will be discussed in the later chapters.

Chapter 3

Problem Statement and Methodologies

To capture the social interactions while evoking natural gestures from people, researchers at CMU created a game for volunteers called the Haggling Game. The experiment was conducted at the CMU Panoptic studio with the setup as shown in the Figure 3.1. The protocol of this game is as follows:

3.1 Haggling Game

In this game, there is a buyer and two sellers. The sellers are trying to sell their products to the buyer and the buyer has to decide at the end which one to go with. In the experiment each game lasted for 1 minute and the seller whose product was bought was given $5. The volunteers were randomly divided into groups of 3 and had the rules of the game explained to them before the start. They were also asked to spend time inside the studio to get familiar with it. The roles were assigned randomly and the sellers were given information about the product they were selling a minute before the game started. There were no instructions provided to the participants about how to position or move around.

During the game, the researchers recorded the position, orientation, voice and body motions. A signal was sent to indicate the start and end of the game and the buyer at the end would write down which seller they were going to buy from. This game was

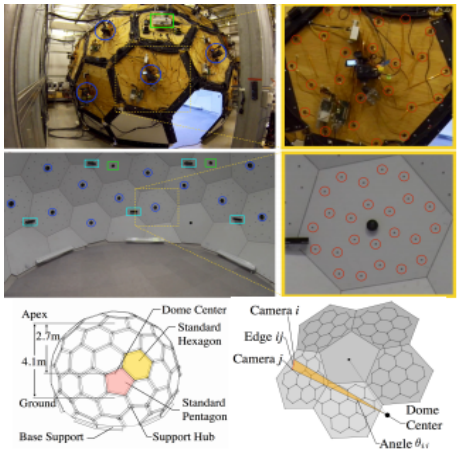

Figure 3.1: The studio structure. (Top) The exterior of the dome with the equipment mounted on the surface. (Middle) The interior of the dome. Red circles show VGA cameras, blue circles are the HD cameras, cyan rectangles the Kinects, and green rectangles are the projectors. (Bottom left) The panels are designed to ensure interchangeability. (Bottom right) Optimized camera positions to ensure uniform angles with respect to the dome center between each camera and all its neighbors (e.g., Camera i is a neighbor of Camera j).Source: [1]

selected as it prompts spontaneous reactions from the players. It can lead to a variety of interactions depending upon the players which the authors claimed was missing from previous datasets capturing dyadic interactions.

3.2 Objective

The objective here is to predict the body gestures related to the behavior in a social situation. Given a scenario of triadic social interaction as described above in the haggling game we consider the social cues of the buyer and one of the sellers and try to predict the same for the other seller. The authors hypothesized that the gestures of one person would be dependent on the other two in the scene and introduced three challenges:

- To predict social formation which would include predicting position and orientation of the person of interest.
- Predicting speaking status which would indicate if the person we are predicting for is speaking or not based on the behavior of the people in the scene.

- Predicting body gestures which involves predicting kinesic signals such as the movement of arms and legs from the movements of the other people.

In our experiments, we are focusing on the third challenge and in the further chapters we will explore how the body gestures have been represented so that they can be utilised as information to our models.

3.3 Methodologies

In this section we explore various techniques and architectures which inspired the models for our experiments. We first introduce some networks which are an integral part of each model discussed below and then talk about the baseline method which the authors of the dataset used and then about MT-VAE and MVAE models from which our models have originated.

3.3.1 Convolutional Neural Networks

A neural network is a type of architecture used to emulate a simplified version of the human brain. As various neurons send and receive messages in the human nervous system, a neural network also connects various functions(called neurons) to form a network capable of sending, receiving and processing information to learn the outcome of the data provided to it. It is a very popular machine learning algorithm. It consists of an input layer, hidden layers and an output layer. The input and output layers are determined based on the type of data provided and the type of outcome expected. The hidden layers can be created of any size and can be as many in number as the user wants. Each layer receives weighted input from the previous layer and applies an activation function(a nonlinear function like sigmoid, ReLU etc.) and sends the output to the next layer. A neural network is trained by defining a loss function based on its output and the provided one and backpropagating [17] this loss through each layer.

A convolutional neural network(CNN) [18] was designed such that the weights are shared by the neurons in the same layer. Instead of every neuron in one layer being connected to every neuron in the next layer with a different weight, the neurons at some interval have the same weight with which their output is multiplied with. This

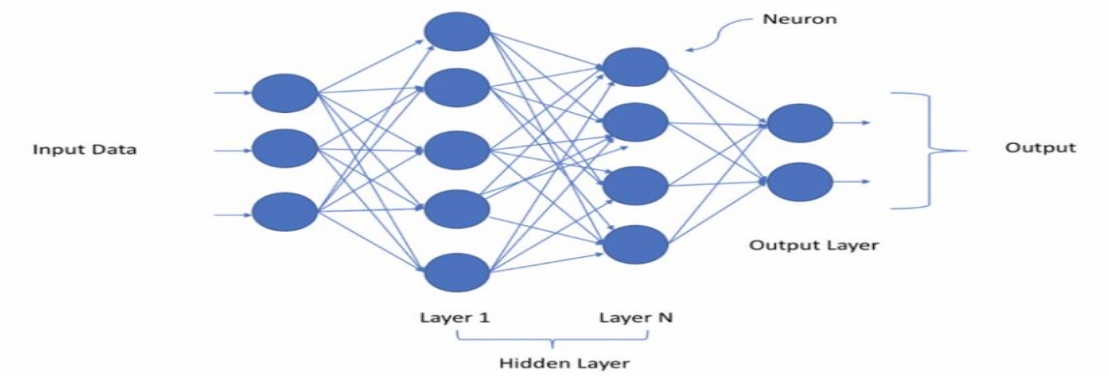

Figure 3.2: An example of a Neural Network. Source:[2]

is the same as a small kernel being applied to sets of neurons with a particular stride. The operation of applying the kernel to a window of neurons is called the convolution operation in which the output of a neuron is multiplied with the corresponding weight in the kernel. Usually a convolution operation is followed up by a pooling layer which collects values from the neighbors in a defined window size and applies a function on the values. Dense layers also used in CNN which are nothing but the same fully connected layers used in neural networks as we described above. CNNs are very popular for image processing and pattern finding tasks. They also help reduce the number of parameters in neural networks. An example of CNN is as shown in the Figure 3.3.

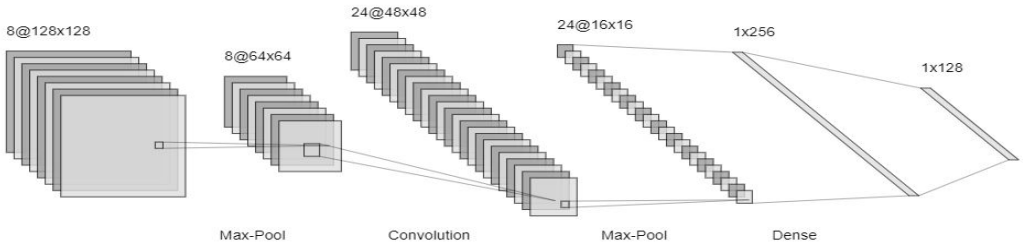

Figure 3.3: An example of a Convolutional Neural Network

3.3.2 LSTM

Long Short Term Memory(LSTM) [19] Networks are a type of recurrent neural networks which were introduced after Recurrent Neural Networks(RNNs) [20] struggled to deal with long term information. RNNs are a type of network which contains a repeating block of network which can apply information learned previously to process current input. They are very useful in dealing with sequential data and have been used extensively in solving language related tasks such as text translation from one language to another where knowledge of previously occurring words is crucial to determining the next word. An example of RNN is shown in the Figure 3.4.

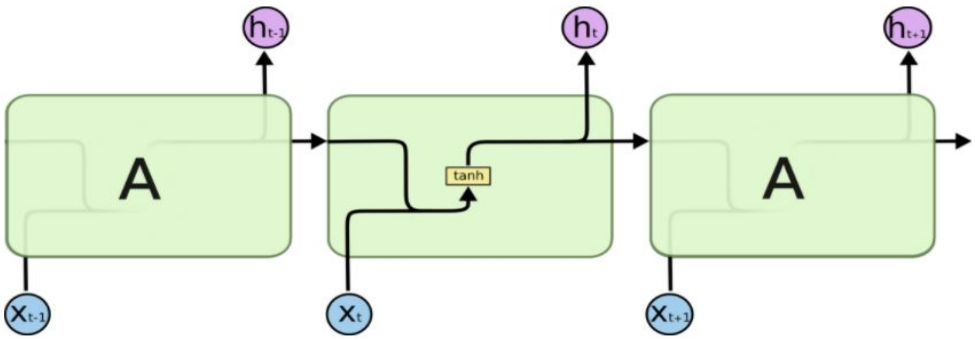

Figure 3.4: An example of RNN. Source:[3]

The problem with RNN is that as the length of sequence grows they start to forget the data learned much earlier and hence do not remain very useful in case of long sequences. That is where LSTMs come in. Their primary characteristic is to remember long term information. LSTMs also have a repeating module but instead of having just a single function, they have four different functions which determine how much information from a particular module will go ahead and how much previous information will be retained.

3.3.3 Autoencoders

Autoencoders [21] are neural network based architectures which learn to reconstruct their input. An autoencoder has two parts - an encoder and a decoder and they can be any type of neural networks, even CNNs or LSTMs. The job of the encoder is to take in

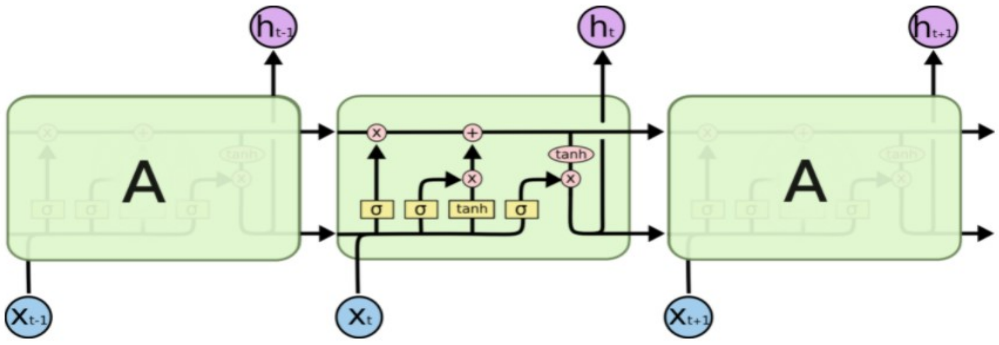

Figure 3.5: An example of LSTM. Source:[3]

the input(X) and map it into a hidden dimension(z). The decoder takes in the output of the encoder(z) and reconstructs it back to the input given(X). The autoencoder function can be represented by the equations :

$$\mathbf{z} = \mathbf{F}_{\mathbf{X}\to\mathbf{z}}(\mathbf{X})$$
$$\mathbf{X}^* = \mathbf{G}_{\mathbf{z}\to\mathbf{X}}(\mathbf{z})$$

The autoencoder is trained such that its output(X^*) is as close to the input(X) as possible. So the loss function while training autoencoders is defined as $L = \|X^* - X\|_2$

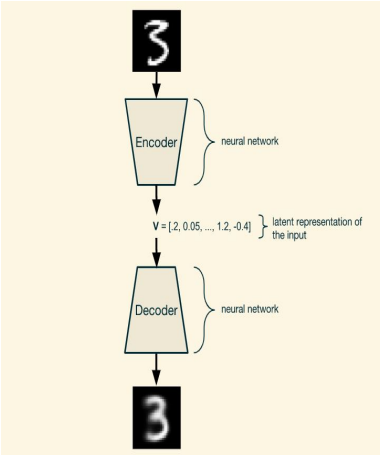

Figure 3.6: A standard autoencoder. Source:[4]

3.3.4 Variational Autoencoders

Variational Autoencoders (VAEs) [22] make use of the same principle as autoencoders but instead of encoders producing specific values of the hidden parameters, they output a probability distribution. The distribution is such that if we sample points from it(z) and pass it through the decoder, we get a duplicate of the input as output. The latent distribution output by the encoder is a gaussian of the same dimension as the latent dimension. The decoder takes in a point sampled from this distribution as input and produces the output belonging to the same class as input. While testing we want to sample points from a distribution like standard normal which the decoder can take and return a meaningful output. So while training we train the encoder such that the sum of distributions from it emulates a standard normal distribution.

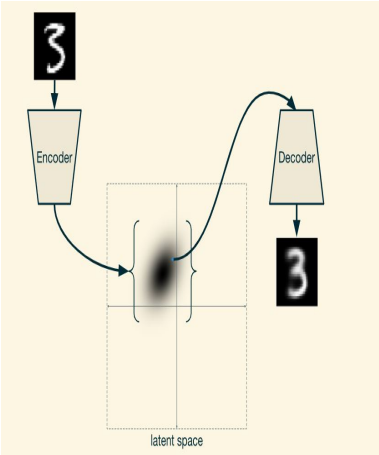

Figure 3.7: A standard VAE. Source:[4]

To make the distributions from the encoder as close to the standard normal as possible. Kullback-Leibler divergence [23] also known as KLD, can help us measure how close our distribution is to the target. Therefore if we want the encoder distribution $Q(z|X)$ to approximate the desired distribution $P(z|X)$, we add the following to our loss function: $D_{KL}(Q(z|X)||P(z|X)) = E(\log(Q(z|X)) - \log(P(z|X)))$

3.3.5 Conditional VAE

In a VAE, we can sample any point and obtain a random output which, if our data could be separated into different classes, belongs to one of the classes. We cannot generate output for any class we desire. That's where Conditional VAEs [24] come in. We feed the encoder with a condition (c) which could be the class of the input or any constraint which could bring out certain features in the output and the same condition is fed to the decoder as well while constructing the output. The encoder learns to encode information other than the condition provided in the hidden dimension. It could encode information about the input which varies with the condition. So the decoder learns to mold the parameters of the hidden dimension using the condition provided into the required output. Hence, the KLD loss function changes to: $D_{KL}(Q(z|X,c)||P(z|X,c)) = E(\log(Q(z|X,c)) - \log(P(z|X,c)))$

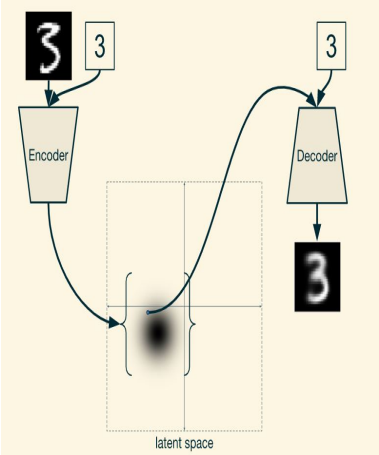

Figure 3.8: A standard conditional VAE. Source:[4]

3.3.6 Baseline Method

In [8], the authors first trained an autoencoder over the data in Holden representation(discussed in the next chapter) ($g : \mathbb{R}^{frames*73} \rightarrow \mathbb{R}^{e}, h : \mathbb{R}^{e} \rightarrow \mathbb{R}^{frames*73}$). It maps the input into a latent dimension and the decoder converts it back into original representation. The trained decoder from this network is kept to convert the motion produced in the latent dimension back to input representation. The decoder is appended after a

convolutional block which processes the poses of other two subjects i.e. the buyer and the other seller. The network architecture for the autoencoder (bodyAE) and prediction model (BodyMotion) are as shown in the Figure 3.9.

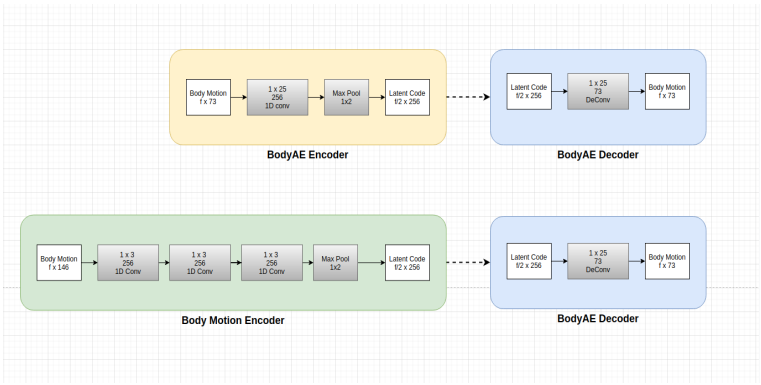

Figure 3.9: The baseline architecture. BodyAE is the encoder and decoder trained initially to map to latent dimension. Body Motion works on the input and outputs in the latent dimension which is then converted to required form by the BodyAE decoder.

3.3.7 MT-VAE

This model was introduced by [5]. This model builds on the idea of using a simple VAE (vanilla VAE) for pose generation. A sequence of poses along with the environmental conditions is input into a VAE and the decoder tries to reproduce the pose.

To further improve the vanilla VAE the authors came up with a new model. They encode the motion sequence into a latent dimension for both target and input. The MTVAE has four components : 1) A LSTM encoder mapping the inputs and target sequences into motion features ($f : \mathbb{R}^{frames*input\_dim} \rightarrow \mathbb{R}^{e}$). 2) A latent encoder combines the motion features (e_a and e_b) and computes the latent feature ($h_{t \rightarrow z}$) 3) a decoder decodes the latent features and input motion features to get target motion features ($h_{z \rightarrow t}$) which are then converted to output sequences by 4) a LSTM decoder ($g : \mathbb{R}^{e} \rightarrow \mathbb{R}^{frames*input\_dim}$). It is shown in the Figure 3.11 The motion features of the input and the target are combined with an additive transformation as shown in the figure : . Initially the input motion features are subtracted from the target features and this is passed on to the latent encoder. Once the points are sampled from the latent space the motion features of the input are concatenated to them and decoded. These

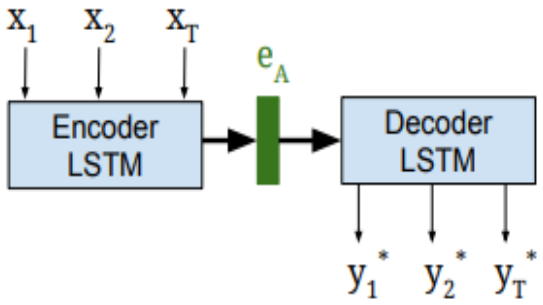

Figure 3.10: LSTM encoder and decoder. The decoder trained here is used to convert the output of VAE used ahead into the input shape. Source:[5]

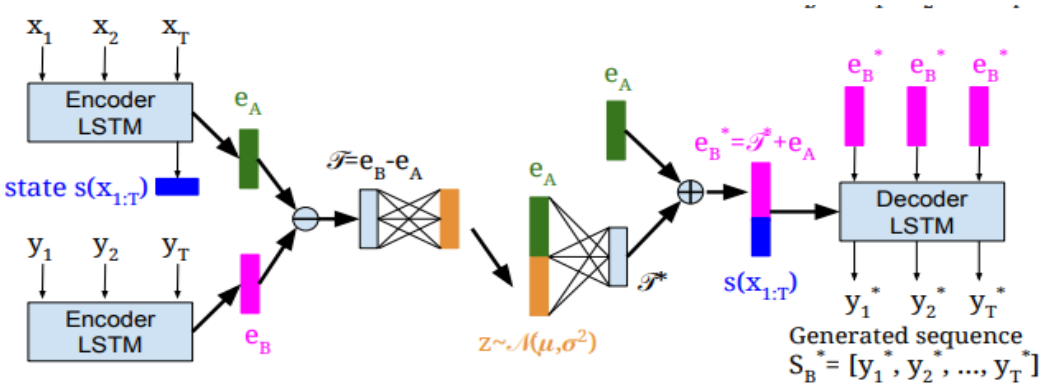

Figure 3.11: Model architecture for MT-VAE. Source:[5]

features are also added back into the output of VAE decoder to finally decode the future sequence as seen in the figure.

3.3.8 MVAE

In the previous approach we made a VAE try to generate motion features from the latent features. The decoder of the VAE would take in the latent features and produce the motion features for the target subject. But one network in the decoder may not perform as well as multiple networks. Therefore, they chose to work with a model used in [6] to generate motion as shown in Figure 3.12.

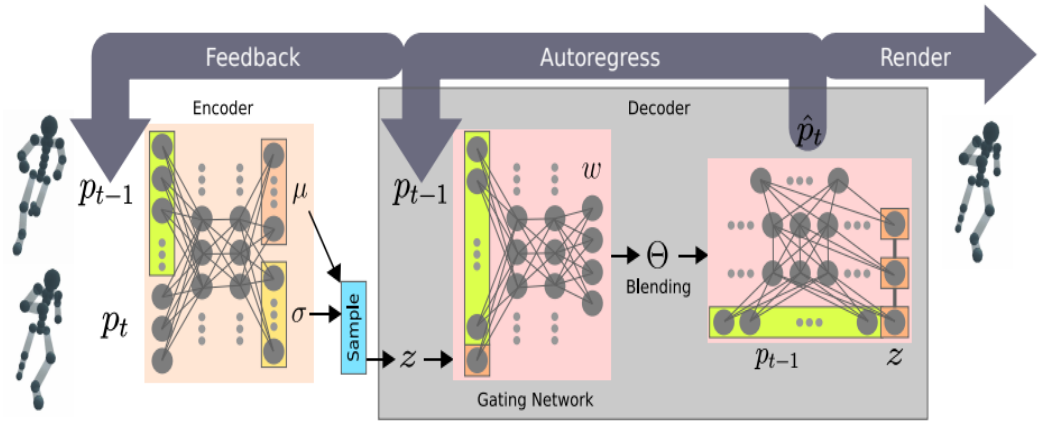

Figure 3.12: VAE architecture used to build character controllers. Source:[6]

They follow the Mann representation as discussed earlier and use a three layer feed-forward neural network as encoder. They encode the pose at the current time t (p_t) and the pose at time t-1 (p_{t-1}) into latent dimension z. The output is divided into two layers to μ and σ for the reparameterization trick for training VAEs [22]. The number of dimensions in the latent space is 32.

To decode the sampled point from the latent dimension, they use a mixture of experts [25] model architecture which was inspired by [26]. Each expert is a neural network that decodes the latent variable z and a gating network blends the weights of the experts. Simply put it gives weightage to each expert and decides how much each expert contributes to the output. Both the gating network and the decoder experts are similar to the structure of the encoder network.

Chapter 4

Dataset and Preprocessing

Using the haggling game dataset of 180 sequences spanning over 3 hours in total was collected. Each sequence contains the detailed features about all three participants referred to as buyer, leftseller and rightseller (The left and right directions are from the perspective of the buyer). To capture the dataset, over 500 cameras for recording videos, 10 RGB+D sensors for depth maps and 23 microphones for sound were synchronized and used. Fusing the depth maps provided the 3D point clouds of each individual. The speech values are represented by a boolean for each frame indicating if the subject is speaking or not. Some features for extracting facial characteristics were provided. As we are dealing with 3D body positions and speech data in our experiments they are discussed in detail below.

4.1 Human Skeleton Representation

The human body is an intricate structure and it is hard to represent every part of the body with 3D points. Hence the human body in the data is represented by 19 joints in the body which makes it easier to capture motion of the body as well as the social cues. This representation is as shown in the Figure 4.1. Although this representation is very effective in portraying the human motion but it doesn't have toes and it cannot capture footstep signals which are used as features for training. The footstep signals are obtained from the bone connecting toe and ankle and let us know if the feet of the person are mid-air, on the ground or in motion.

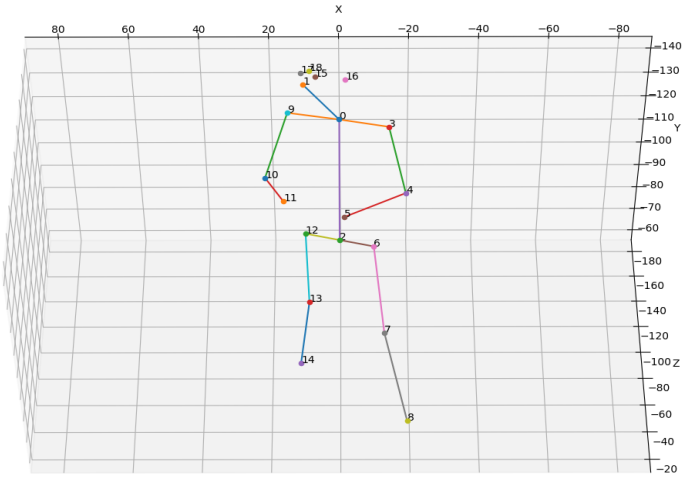

Figure 4.1: Human Skeleton using 19 joints

4.2 Retargating Skeleton

To train our models we need to convert the 3D body positions into a suitable format. We also need to extract more information from the pose that may provide us social cues helpful for training. The 19 joint representation is retargeted to 21 joint representation as described in Holden et al [14]. The joint positions are mapped to their corresponding positions in the new representation. A base skeleton is taken, scaled and using an inverse kinematic solver, it is transformed into the correct pose. The new skeleton adds a point for each toe connecting the ankle. We then calculate the forward direction using the vectors across the shoulders and hips and averaging them and taking a cross product with y-axis. The translational as well as the rotational velocity of the subject(of the root(pelvis) joint in the pose) is calculated in each frame as well. The sequence is broken down into windows of fixed length (120 frames in each window for our experiments) with a little overlap (10 frames) between consecutive windows for training. The new skeleton representation is shown in Figure 4.2.

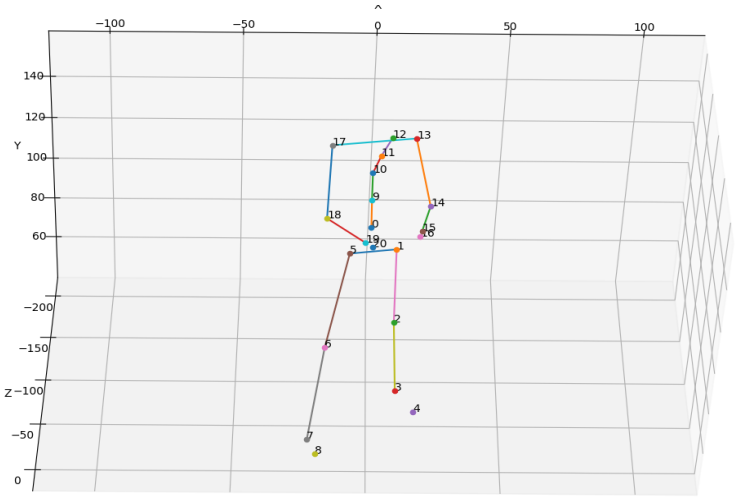

Figure 4.2: Human Skeleton using 21 joints

4.3 Holden Representation

Now that the body positions have been calculated, some other features are extracted to give more information about the social interaction. These are mainly related to the position and orientation of each subject in their social formation.

The process described in Holden et al [14] is as follows : First the root (pelvis) projection on the ground is taken. The foot contact signals are generated and the velocity of the root joint for each frame in the window is calculated. The skeletons are rotated along the y-axis so that they face the forward direction. This was done so that the model would learn to produce body movement and not fixate on producing positions. These rotated joint positions along with the angular and translational velocity of the root and foot contact signals are combined to form a 73 dimensional feature for each frame of the window for each subject. The information about the rotation and translation not included in this representation is saved to obtain back the skeleton from this representation.

4.4 Mann Representation

Another feature representation we used in our experiments was introduced by Mann et al [6]. They use a 244 dimensional feature representation to train their model which is extracted as follows :

Some of the features extracted are similar to Holden representation. We get the root projection onto the ground and calculate the forward direction (orientation) of the body and project it onto the ground. We also calculate the root angular and translational velocities. We then project the joint positions in the frame of root projections and calculate joint translational velocities in this space. This is done so that our model generates results regardless of which seller it is producing the result. Also, joint orientations, which are vectors from one joint to another in the skeleton (bone connecting two parts), are calculated and projected in the y and z directions.

All the joint positions, velocities and orientation and the root angular as well as translational velocities are concatenated to form a 244 dimensional feature vector.

Chapter 5

Experiments and Analysis

In this chapter we explore the metrics we use for evaluating and comparing our experiments and also provide the results of our experiments. The results can be evaluated both quantitatively and qualitatively. The first two metrics explored here are for the quantitative evaluation and the last topic caters to the qualitative evaluation of the results. We also explore the implementation of the baseline model and the model architectures we researched and came up with to beat the baseline. In each model we use the pose representation of other two subjects to predict the pose of a seller.

5.1 Implementation Details and Training

We implemented each model mentioned in the methodologies and those details and the design inspired from those models are discussed below.

5.1.1 Baseline Model

We use the same model as the one described by the authors as shown in Figure 3.9. At training time, the model takes in the sequence of poses of the buyer (b) and the other seller (s_2) in the Holden representation (frames*2*num_dimensions, here it is 120*2*73), passes them through the convolution and max pooling layers and uses the previously trained bodyAE decoder to output the sequence of poses of the target seller (s_1).

$$s_1(\mathbf{t}_0 : \mathbf{t}) = \mathbf{F}_{b,s_2 \to s_1}(b(\mathbf{t}_0 : \mathbf{t}), s_2(\mathbf{t}_0 : \mathbf{t}))$$

The baseline methods use MSE loss as their training loss and also as the only evaluation metric.

5.1.2 MT-VAE

Our model uses a similar structure. We replace the LSTM encoder and decoder with a convolutional encoder and decoder we used in the previous experiment as convolutional autoencoder better helps extract features from the other two subjects. The input features x (frames*input_feature_dim*2) which is 120*244*2 for Mann and 120*73*2 for Holden representation are passed through the convolutional encoder to get e_a. The target features y converted to e_b are of shape (frames*input_feature_dim). The latent encoder receives $t = e_b - e_a$ and transforms it into latent space via $z = h_{t \to z}(t)$. The latent decoder reconstructs the feature t^* using $h_{z \to t}(z|e_a)$. Because we subtracted the input motion features before entering into VAE, we add them back before passing through the convolutional decoder $y^* = g(t + e_a)$.

To train the model, we used three different kinds of losses. The first is the MSE loss function as used in the previous model between the generated and ground truth pose representations of the target. The second is KL-divergence loss for training the VAE. As discussed earlier, it is to get the latent dimension from which the point is sampled from as close to standard normal as possible. The third is the cycle consistency loss which was used by the authors of MT-VAE. We get the latent dimension z while training and the latent decoder gives us the output e_b^*. The e_b^* is put back into the VAE encoder and z^* is obtained. The cycle consistency loss is defined as follows :

$$\mathbf{L} = \|\mathbf{z}^* - \mathbf{z}\|_2$$

where $z^* = h_{t \to z}(h_{z \to t}(z, e_a))$.

This is done to regularize the feature space. We have MSE loss for the generated output to ensure the model learns the information as intended, KLD loss to ensure it learns to sample from standard normal and cycle consistency loss to ensure regularization in latent space. From the experiments this cycle is selected and not a bigger cycle consisting of end pose generated being input because it was found ineffective. It might have been due to exploding or vanishing gradients. Hence The loss function for this

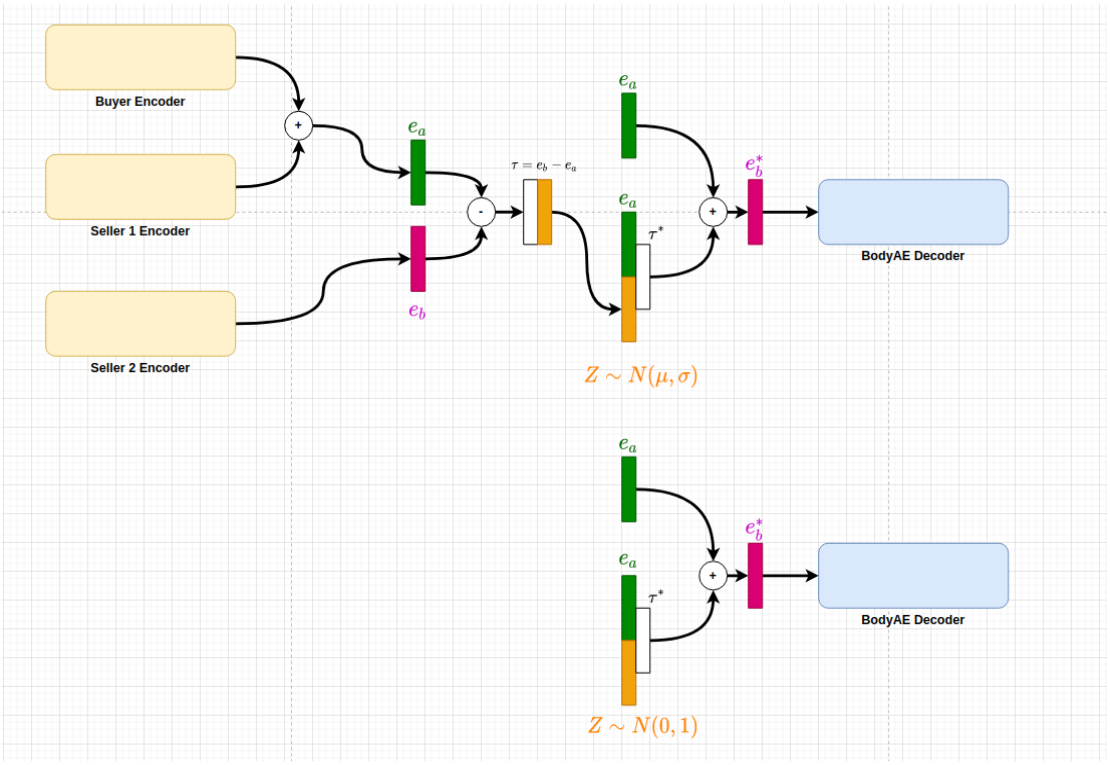

Figure 5.1: Our Model architecture inspired from MT-VAE.(Top) Model architecture while training (Bottom) Model architecture used while testing

model is as follows:

$$\mathbf{L} = \|\mathbf{y}^* - \mathbf{y}\|_2 + \lambda_{KLD}\mathbf{D}_{KL}(\mathbf{Q}(\mathbf{z}|\mathbf{t})||\mathbf{P}(\mathbf{z}|\mathbf{t})) + \lambda_{cycle}\|\mathbf{z}^* - \mathbf{z}\|_2$$

where $P(z|t)$ is the standard normal $N(0, 1)$ and λ_{cycle} and λ_{KLD} are the regularization factor for the cycle consistency and KL-divergence loss. Once the model has been trained, we can discard the encoder of VAE of the model. We can simply sample a point from the standard normal for z and combine it with e_a to get the output as shown in the bottom half of Figure 3.3.7

5.1.3 MVAE

To adapt this model to our problem statement, we made a few changes to the input. Since we are predicting one of the sellers, we give the current pose in mann features

(s_{2t}) and its pose in previous frame (s_{2t-1}) as input. Along with it, the buyer pose (b_t) and the other seller pose (s_{1t}) in the current frame are given as input. To give more information about the context, we also decided to input the speaking statuses of the subjects (c_t) in the current frame as input as well. The features s_{2t-1}, s_{1t}, b_t and c_t comprise the condition C for our VAE and fed along with the latent state z to the decoder. The architecture can be seen in the Figure 5.2. We used an encoder with 3 dense layers and 2 layers to calculate the mean and standard deviation. In the decoder network, we used a mixture of 8 experts and the corresponding gating network to blend their weights. We also used teacher forcing [27] for some part of the training. When activated, we used the ground truth seller pose in the previous frame as input otherwise we use the pose generated by the model for the current frame as input to the next frame. The decoder outputs the generated output s^*_{2t} and if activated this output is passed through a linear layer which gives an output determining if the seller is speaking or not (c^*).

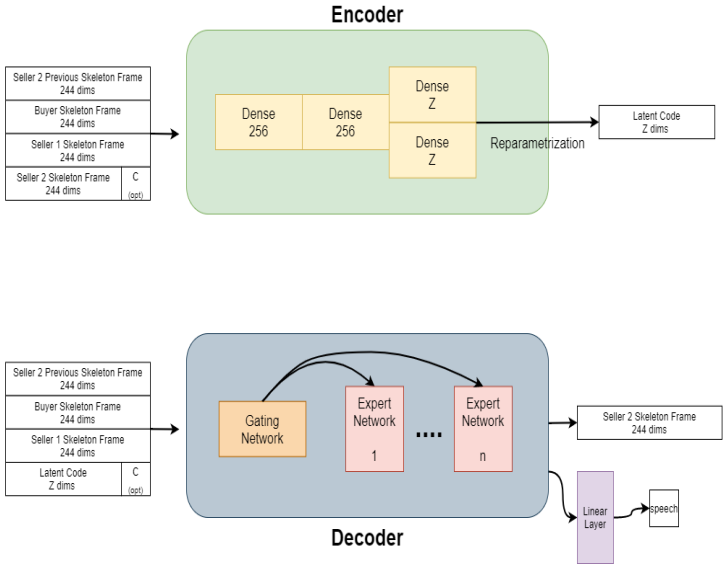

Figure 5.2: MVAE architecture. (Top) Shows the encoder which converts input into the latent dimension. (Bottom) Shows the Decoder architecture. It contains the Gating network which blends the Mixture of Experts and gives the output in the required shape. The gating and expert architecture is shown in Figure 5.3

For the loss function, we use the MSE and KLD losses and cross entropy loss if the model also predicts the speaking status of the subjects. It is written as :

$$\mathbf{L} = \|\mathbf{s}^*_{2t} - \mathbf{s}_{2t}\|_2 + \lambda_{KLD}\mathbf{D}_{KL}(\mathbf{Q}(\mathbf{z}|\mathbf{s}_{2t}, \mathbf{C})||\mathbf{P}(\mathbf{z}|\mathbf{s}_{2t}, \mathbf{C})) + \lambda_{speech}\mathbf{B}(\mathbf{c}^*, \mathbf{c})$$

Here λ_{speech} is regularization factor for speech loss $B(c^*, c)$ which is the binary cross entropy loss defined as $B(x, y) = y\log(x) + (1-y)\log(1-x)$ and c is the ground truth speaking status of the target seller.

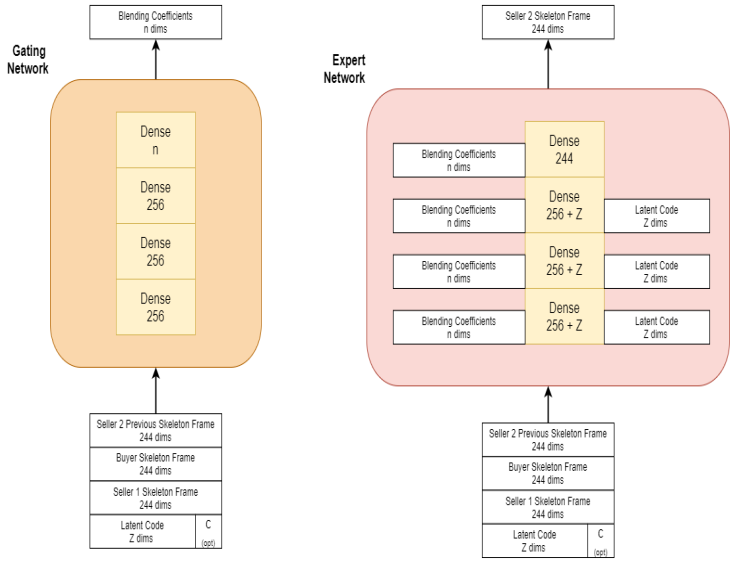

Figure 5.3: MVAE Gating and Expert architecture. (Left) Shows the Gating network. (Right) Shows the architecture of one of the Experts used in the Decoder

Testing this model is the same as testing MT-VAE. Since we have a conditional-VAE model, we can ignore the encoder part. We sample a point from the standard normal and combine it with our other inputs and feed them to the decoder network.

5.2 Hyperparameter Tuning and System Details

The entire code was written using pytorch framework [28]. We used Weights and Biases(wandb) [29], a python library for running hyperparameter tuning. Using the sweeps feature in the wandb library, we set up hyperparameter tuning jobs on our machines(6

GB Nvidia RTX 2060 GPU). The library automatically chooses the values of the parameters to be tuned using bayesian optimization [30] and runs training job on the machines. We used this functionality for training our convolutional autoencoder, Body Motion and MT-VAE models. For the MVAE approach, we did not run hyperparameter tuning jobs, as it is a large network and using a learning rate of 0.001 and regularization parameter of 0.2 was sufficient for it. We trained this model on a 32 GB Tesla V100 GPU available at the Minnesota Supercomputing Institute(MSI). The training time for each model took 4-5 hours except for MVAE with 8 experts which took 3 days to train for 400 epochs and we used Adam optimizer [31] for learning.

5.3 Evaluation Metrics

5.3.1 MSE

This was the metric used by the original authors in [8]. It measures the mean squared error between each joint position in the generated (J^*) and ground truth pose(J). The root joint (pelvis) is aligned for both the poses and the we calculate the mean over the number of joints and frames. The formula is as follows :

$$\mathbf{E} = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\|J_i^* - J_i\|_2$$

5.3.2 Frechet Distance

This metric was used for evaluating body pose generated by the model in [16]. In this paper, the authors worked on generating upper body pose based on audio and text signals and evaluated them based on the Frechet distance [32] between the ground truth body sequence and the generated one. We followed the same process as them. We used an autoencoder which takes in joint directions as input and maps them into a hidden dimension and recovers them back. Joint directions are calculated as the difference of joint positions connected directly in the 3D skeleton. In the 21 joint representation we have 14 such directions. After the autoencoder has been trained, the encoder part is kept for evaluation. We convert the generated (X^*) and ground truth (X) pose features into the joint representation and get the joint directions from them. They are projected

into a latent dimension using the encoder. We calculate the frechet distance between X and X^* as follows :

$$\mathbf{F} = \|\mu_{gt} - \mu_p\|_2^2 + \mathrm{Tr}(\Sigma_{gt} + \Sigma_p - 2(\Sigma_{gt}\Sigma_p)^{\frac{1}{2}})$$

Here μ_{gt} and Σ_{gt} represent the mean and covariance matrix of the ground truth pose directions in latent dimension Z_{gt} and μ_p and Σ_p are the mean and covariance of distribution of generated pose directions in latent space Z_p.

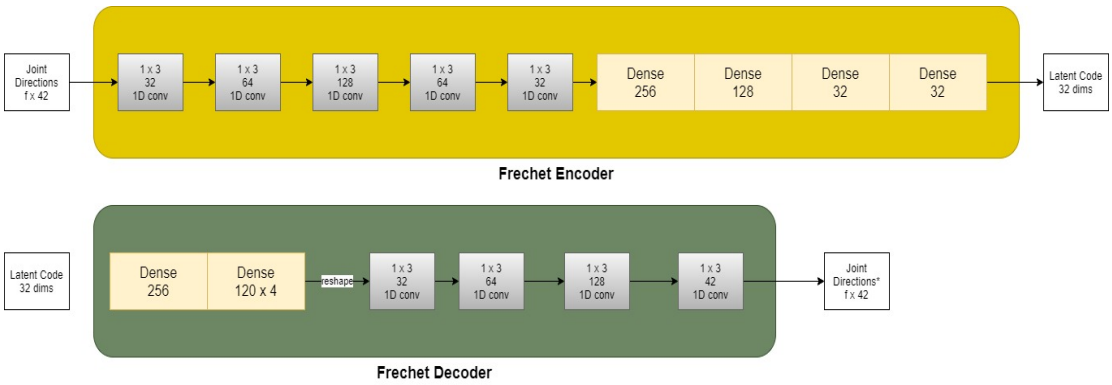

Figure 5.4: Autoencoder network architecture. The Encoder (top) is used to map joint directions into latent dimension which is then used to calculate Frechet distance.

But why frechet distance theoretically works? In the paper [33] Zhang et al., explored that deep neural networks learn to extract out features which are related to visual perception of the images. Hence the features extracted using neural networks would reflect the properties that humans look for while visually differentiating between images or videos. Frechet distance measures how a distribution differs from another. It works for both continuous and discrete distributions. So theoretically, the features extracted from our generated output which are visually closer to ground truth should have a lower frechet distance and thus quantitatively allow us to judge different models.

5.3.3 Visual Inspection

As the authors in [8] pointed out it is hard to measure the correctness of the gestures with mathematical representations because of the lack of rules which quantify the gestures of people. However a combination of MSE and Frechet Distance does show promise

in measuring the correctness of such gestures as we will see in our analysis. But we do need a qualitative analysis of the output in our case. Visual inspection is the only way to determine the quality of output from the experiments. A model may produce good numbers in quantitative evaluation when in reality the output could be just a mean pose which reduces error in measurement but shows no sign of movement which does not happen in most social interactions. Hence visual inspection is a big factor in determining the success of the experiment and we rely on it heavily for evaluation of the model output.

5.4 Analysis

5.4.1 Results

Based on the metrics described in the previous chapter, we present the results of the experiments we performed in the table 5.1 and 5.2. Since we make predictions for both the left seller and the right seller, we also have metrics for each of them. The MSE and Frechet distance is then calculated by taking the average of left and right values.

Metric	Body Motion	MT-VAE	MVAE
MSE	8.897	7.602	5.035
LeftMSE	9.201	8.264	5.069
RightMSE	8.592	6.940	5.002

Table 5.1: Performance of different models on MSE

Metric	Body Motion	MT-VAE	MVAE
Frechet	11.761	8.425	23.595
LeftFrechet	12.617	9.141	18.077
RightFrechet	10.906	7.707	29.113

Table 5.2: Performance of different models on Frechet distance

Theoretically, both MSE and Frechet should be lower in value for a better model. Looking at the values in the tables 5.1 and 5.2 we cannot draw a conclusion on which model performs better. Although it looks like MT-VAE which has lower MSE as well as Frechet should be a better model, but visually the output of MVAE shows more range of motion and looks more natural. This leads us to believe that a proper weightage

should be given to each metric if we want to combine their results. But we cannot give any ratio to each metric which favors us and thus need to research more on this matter. There could be a reason why Frechet distance for Body Motion and MT-VAE would be lower than that for MVAE. Both these models use a convolution based autoencoder to generate features and then work on them just like the autoencoder which extracts features for Frechet. It is possible that the convolution encoder in these models would extract similar features from the input as the encoder of Frechet model and instill them in the output. Whereas the MVAE encoder uses feedforward linear layers to extract the features of input. This would explain the bias of Frechet distance towards Body Motion and MT-VAE. But it show that MT-VAE is a better model than Body Motion which also agrees with us visually.

5.4.2 Ablation Study

We conduct an ablation study on the usefulness of including the speech data in our input to the MVAE model. When we provide input to the model, we have an option of providing the status of each person in the scene about whether they are speaking or not. We can include this information with a 0 or 1 to indicate speaking. We consider the three ways we can use the speech data.

- Provide speaking status of the buyer and other seller.
- Provide speaking status of the seller in the previous frame for which our model is generating the pose.
- Do not provide speaking status of anyone at all.

We modify the dimensions of c in our MVAE architecture to observe the results of this study. The number of dimensions of $c(c_{dim})$ are 2, 1 and 0 for the corresponding cases. The MSE and Frechet distance is measured for each case and presented in the table 5.3.

We see that removing the speech data theoretically deteriorates the performance. The model with $c_dim = 2$ shows the best performance because we provide the most information about the context. When $c_dim = 1$, the model only knows if the target was speaking or not in the previous frame, and thus might learn to generate the output

Metric	MVAE($c_{dim} = 0$)	MVAE($c_{dim} = 1$)	MVAE($c_{dim} = 2$)
MSE	25.664	35.846	23.595
Frechet	5.313	5.657	5.035

Table 5.3: Effect of speaking status on metrics

which agree with the speaking status but maybe not agree with the ground truth body positions. When c_dim is 0, the model relies solely on the body joint positions provided and thus could theoretically be better than the case when c_dim is 1.

5.4.3 Visualization

We present some results of visualization in the Figure 5.5. Since these are still images they just show how we visualized the output along with ground truths. The yellow skeleton show the generated output. They have been overlapped with the ground truth to observe the difference in the ground truth and the output.

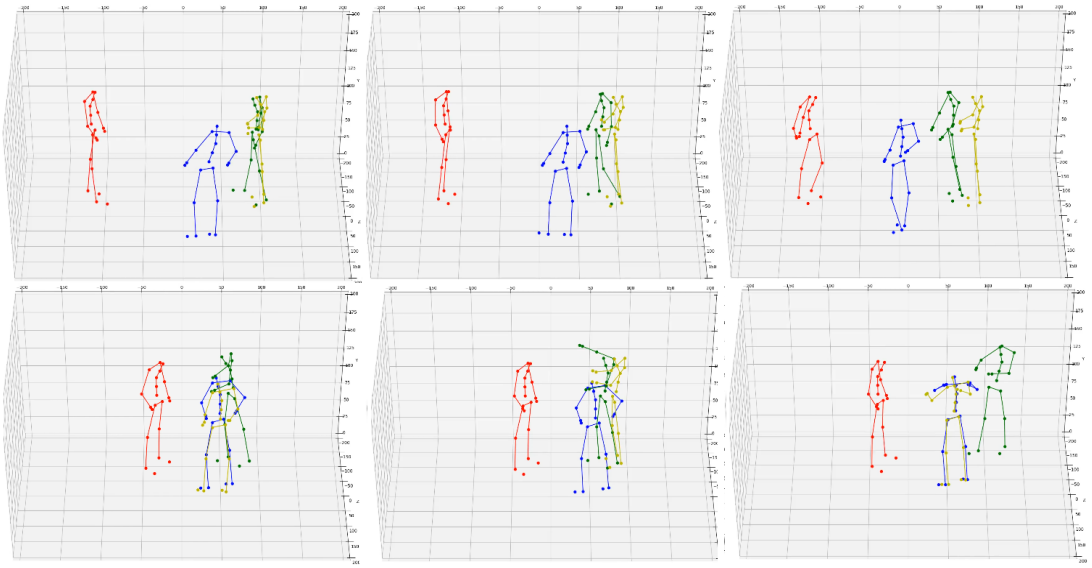

Figure 5.5: Visualization of Output. The characters in green, blue and red are the ground truth body poses and the one in yellow shows the generated output

Chapter 6

Conclusion

We explore the field of Social Artificial Intelligence and experiment with a dataset composed of videos of Triadic Social Interactions. We define frameworks for utilizing the social cues exhibited by the people in social situations and confirm the hypothesis that the social cues of a person are dependent on the social cues of the people they are interacting with. We also looked at the metrics using which we can determine the efficacy of our models although we still need to conduct more research to confirm that the metrics are appropriate for our purpose.

In the future we plan to use Frechet evaluation based on other models like LSTM which can serve as unbiased estimators of the performance of social gesture generation models. We also aim to look at new methods like attention models to possibly generate more visually appealing and realistic outputs. There is also a need for other evaluation metrics which can measure the naturalness of human gestures in a given context. All in all, this is a stepping stone for research in social gesture generation models and we believe the future holds many more exciting discoveries in this field which would revolutionize human robot interaction.

References

[1] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic Studio: A Massively Multiview System for Social Interaction Capture. *arXiv:1612.03153 [cs]*, December 2016. arXiv: 1612.03153.

[2] Jojo John Moolayil. A Layman's Guide to Deep Neural Networks, May 2020.

[3] Understanding LSTM Networks – colah's blog.

[4] Conditional Variational Autoencoders.

[5] Xinchen Yan, Akash Rastogi, Ruben Villegas, Kalyan Sunkavalli, Eli Shechtman, Sunil Hadap, Ersin Yumer, and Honglak Lee. MT-VAE: Learning Motion Transformations to Generate Multimodal Human Dynamics. *arXiv:1808.04545 [cs, stat]*, August 2018. arXiv: 1808.04545.

[6] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. Character controllers using motion VAEs. *ACM Transactions on Graphics*, 39(4), July 2020.

[7] Sharice Clough and Melissa C. Duff. The role of gesture in communication and cognition: Implications for understanding and treating neurogenic communication disorders. *Frontiers in Human Neuroscience*, 14:323, 2020.

[8] Hanbyul Joo, Tomas Simon, Mina Cikara, and Yaser Sheikh. Towards Social Artificial Intelligence: Nonverbal Social Signal Prediction in a Triadic Interaction. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10865–10875, Long Beach, CA, USA, June 2019. IEEE.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020. arXiv: 2005.14165.

[10] Evonne Ng, Shiry Ginosar, Trevor Darrell, and Hanbyul Joo. Body2Hands: Learning to Infer 3D Hands from Conversational Gesture Body Dynamics. *arXiv:2007.12287 [cs]*, April 2021. arXiv: 2007.12287.

[11] Evonne Ng, Donglai Xiang, Hanbyul Joo, and Kristen Grauman. You2Me: Inferring Body Pose in Egocentric Video via First and Second Person Interactions. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9887–9897, Seattle, WA, USA, June 2020. IEEE.

[12] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, 2014.

[13] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *3D Vision (3DV), 2017 Fifth International Conference on*. IEEE, 2017.

[14] Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics*, 35(4):1–11, July 2016.

[15] Anand Gopalakrishnan, Ankur Mali, Dan Kifer, Lee Giles, and Alexander G. Ororbia. A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Youngwoo Yoon, Bok Cha, Joo-Haeng Lee, Minsu Jang, Jaeyeon Lee, Jaehong Kim, and Geehyuk Lee. Speech gesture generation from the trimodal context of text, audio, and speaker identity. 39(6), November 2020.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, December 1989.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[20] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[21] G. E. Hinton. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

[22] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114 version: 10.

[23] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[24] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[25] Tara Baldacchino, Elizabeth J. Cross, Keith Worden, and Jennifer Rowson. Variational Bayesian mixture of experts models and sensitivity analysis for nonlinear dynamical systems. *Mechanical Systems and Signal Processing*, 66-67:178–200, 2016.

[26] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics*, 37(4):1–11, August 2018.

[27] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[30] Julien-Charles Lévesque, Christian Gagné, and Robert Sabourin. Bayesian Hyperparameter Optimization for Ensemble Learning. *arXiv:1605.06394 [cs]*, May 2016. arXiv: 1605.06394.

[31] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017. arXiv: 1412.6980.

[32] Fréchet distance, March 2021. Page Version ID: 1014574383.

[33] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, Salt Lake City, UT, June 2018. IEEE.

Appendix A

Acronyms

Care has been taken in this thesis to minimize the use of jargon and acronyms, but this cannot always be achieved. This appendix contains a table of acronyms and their meaning.

A.1 Acronyms

Table A.1: Acronyms

Acronym	Meaning
AI	Artificial Intelligence
NPSS	Normalized Power Spectrum Similarity
CNN	Convolutional Neural Network
VAE	Variational Autoencoder
KLD	Kullback-Leibler Divergence
MSE	Mean Squared Error